## REST APPLICATION DEVELOPMENT USING SPRING BOOT AND JPA

**Overview:** This presentation explores key annotations provided by Spring Boot and Spring to simplify REST application development using Java. The focus is on core annotations, data access annotations, testing annotations, AOP (Aspect-Oriented Programming) annotations, and Spring Web annotations.

**Core Annotations:**

* **@SpringBootApplication:**
  + Serves as the entry point for a Spring Boot application.
  + It’s a meta-annotation combining @Configuration, @EnableAutoConfiguration, and @ComponentScan.
* **@Controller:**
  + Defines a class as a controller in Spring MVC architecture.
  + Handles HTTP requests and typically returns view names for rendering HTML views.
* **@RestController:**
  + Specializes @Controller for RESTful web services.
  + Handles HTTP requests and directly returns data (e.g., JSON or XML) without view rendering.
* **@Service, @Repository, @Component:**
  + Stereotype annotations for defining different types of Spring components.
  + All indicate a class is a Spring bean, but used in different application layers to express intended purpose.
  + **@Service:** For service classes in the business logic layer.
  + **@Repository:** For DAO classes in the persistence layer.
  + **@Component:** Used as a generic stereotype when more specific ones are not applicable.

**Dependency Injection Annotations:**

* **@Autowired:**
  + Automatically injects dependencies into Spring beans.
  + Supports injection into fields, constructors, setter methods, and configuration methods.
* **@Qualifier:**
  + Used with @Autowired to specify which bean should be injected when multiple candidates of the same type exist.

**Data Access Annotations:**

* **@Entity:**
  + Defines a data entity managed by the ORM (Object-Relational Mapping) system.
  + Maps class fields to database tables/columns and provides CRUD operations.
* **@Repository:**
  + Indicates a class is a Spring Data repository, typically a DAO (Data Access Object).
  + Used to perform CRUD operations on entities and often extends CrudRepository or JpaRepository.
* **@Transactional:**
  + Defines the scope of a single database transaction.
  + Applies to methods or classes to indicate that the annotated method (or all methods in the class) should be wrapped with a transaction.
* **@Query:**
  + Defines a query method using JPQL (Java Persistence Query Language) or native SQL in a Spring Data repository.
* **@Column:**
  + Specifies details of a column in a database table.
  + Used to customize the mapping of a field to a column.

**Testing Annotations:**

* **@RunWith(SpringRunner.class):**
  + Bridges JUnit and the Spring TestContext Framework.
* **@SpringBootTest:**
  + Specifies the configuration of the ApplicationContext in a Spring Boot test.
* **@Configuration:**
  + Indicates a class declares one or more @Bean methods and can be processed by the Spring container.
* **@Value:**
  + Injects values from properties files or other sources into fields.
* **@Conditional:**
  + Conditionally activates a component or configuration class based on the evaluation of a Condition.
* **@Scheduled:**
  + Defines when a method should be invoked based on a fixed rate or cron expression.
* **@EnableAutoConfiguration:**
  + Enables Spring Boot’s auto-configuration mechanism.

**AOP Annotations:**

* **@Aspect:**
  + Identifies a class as an aspect.
  + Contains advice methods and pointcuts.
* **@Before, @After, @Around:**
  + Define advice methods to run before, after, or around a method execution.
* **@Pointcut:**
  + Defines a reusable pointcut expression.

**Spring Web Annotations:**

* **@RequestBody:**
  + Binds the method parameter to the body of the HTTP request.
* **@ResponseStatus:**
  + Sets the HTTP response status code.
* **@ExceptionHandler:**
  + Handles exceptions in a specific controller or globally.
* **@ModelAttribute:**
  + Adds attributes to the model before the handler method is invoked.